(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（Ａ）**　(11)特許出願公開番号

特開平9－330219

(43)公開日　平成９年(1997)12月22日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ			技術表示箇所
Ｇ０６Ｆ	9/30	３１０		Ｇ０６Ｆ	9/30	３１０Ｃ	
	9/45				9/44	３２２Ｆ	

審査請求　未請求　請求項の数14　ＯＬ　（全 15 頁）

(21)出願番号　特願平8－151260

(22)出願日　平成８年(1996)６月12日

(71)出願人　000005821
松下電器産業株式会社
大阪府門真市大字門真1006番地

(72)発明者　高山　秀一
大阪府門真市大字門真1006番地　松下電器産業株式会社内

(72)発明者　槍垣　信生
大阪府門真市大字門真1006番地　松下電器産業株式会社内

(72)発明者　富永　宣輝
大阪府門真市大字門真1006番地　松下電器産業株式会社内

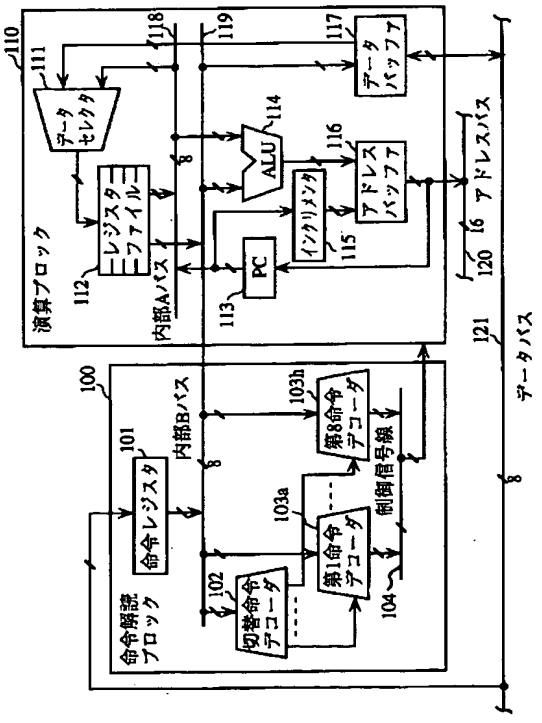
(74)代理人　弁理士　中島　司朗

(54)【発明の名称】　命令解読器切替型プロセッサ及び翻訳装置

(57)【要約】

【課題】　命令語長を増加させずに多くの種類の機械語命令を定義することができるプロセッサを提供する。

【解決手段】　命令解読ブロック１００と演算ブロック１１０とから構成されるプロセッサであって、命令解読ブロック１００はさらに、フェッチした命令を保持する命令レジスタ１０１と、フェッチされた命令を解読し演算ブロック１１０を制御する８個の異なる種類の命令デコーダ１０３ａ～１０３ｈと、フェッチされた専用の切替命令によってそれら８個の命令デコーダ１０３ａ～１０３ｈの中から１個だけを有効とさせる切替命令デコーダ１０２とからなる。



1

【特許請求の範囲】

【請求項１】　命令メモリから命令を読み出して解読する命令解読手段と、その命令解読手段から出力される解読結果に従って演算処理を実行する実行手段とを備えるプロセッサであって、

前記命令解読手段は、

前記命令メモリから命令を読み出すフェッチ手段と、

前記フェッチ手段により読み出された命令を解読し、少なくとも１個の同一命令に対して異なる解読結果を出力するｎ（２以上の整数）個の命令解読器と、

前記ｎ個の命令解読器から選択された一つの命令解読器の解読結果だけが前記実行手段に出力されるよう前記ｎ個の命令解読器を制御する制御手段とからなることを特徴とする命令解読器切替型プロセッサ。

【請求項２】　前記制御手段は、

前記フェッチ手段により読み出された命令が所定の命令解読器切替命令であるかどうかを判断する判断部と、

前記判断部により所定の命令解読器切替命令であると判断された場合には、その命令解読器切替命令に基づいて前記選択を行う選択部とからなることを特徴とする請求項１記載の命令解読器切替型プロセッサ。

【請求項３】　前記選択部は、

前記判断部により所定の命令解読器切替命令であると判断された場合には、その命令解読器切替命令から一の命令解読器を特定する情報を抽出する抽出部と、

前記抽出部により抽出された情報を記憶する記憶部と、

前記ｎ個の命令解読器のうち前記記憶部に記憶された情報に対応する命令解読器の解読結果だけが前記実行手段に出力されるよう前記ｎ個の命令解読器を制御する出力制御部とからなることを特徴とする請求項２記載の命令解読器切替型プロセッサ。

【請求項４】　前記出力制御部は、

前記ｎ個の命令解読器の解読結果が前記実行手段に出力されることを命令解読器ごとに許可又は禁止するゲート部と、

前記記憶部に記憶された情報に基づいて前記ゲート部を制御するゲート制御部とからなることを特徴とする請求項３記載の命令解読器切替型プロセッサ。

【請求項５】　前記ｎは２であることを特徴とする請求項４記載の命令解読器切替型プロセッサ。

【請求項６】　プログラムを翻訳することにより、ｎ（２以上の整数）個の異なる種類の命令解読器を備えるプロセッサを対象とする機械語命令列を生成する翻訳装置であって、

最適な機械語命令列を生成するためには前記ｎ個の命令解読器の中のいずれを対象とすべきかを決定する決定手段と、

前記決定手段により決定された命令解読器を対象として前記プログラムを翻訳し機械語命令列を生成する生成手段とを備えることを特徴とする翻訳装置。

2

【請求項７】　前記決定手段は、生成された機械語命令列のコードサイズが最小となる場合を最適とすることを特徴とする請求項６記載の翻訳装置。

【請求項８】　前記決定手段は、生成された機械語命令列を前記プロセッサが実行するのに要する時間が最小となる場合を最適とすることを特徴とする請求項６記載の翻訳装置。

【請求項９】　前記翻訳装置はさらに、前記プログラムを所定の基本単位に分割する分割手段を備え、

前記決定手段及び前記生成手段は、前記分割手段により分割された基本単位ごとに前記決定及び前記生成を行うことを特徴とする請求項７又は８記載の翻訳装置。

【請求項１０】　プログラムを翻訳することにより、ｎ（２以上の整数）個の異なる種類の命令解読器を備えるプロセッサを対象とする機械語命令列を生成する翻訳装置であって、

前記プログラムから前記ｎ個の命令解読器のそれぞれを対象とする機械語命令列とその最適化に必要な情報とを生成する暫定翻訳手段と、

前記暫定翻訳手段により生成された前記情報に基づいて前記ｎ個の機械語命令列の中から最適な１個を決定しその機械語命令列をこの翻訳装置が生成する機械語命令列とする決定手段とを備えることを特徴とする翻訳装置。

【請求項１１】　前記暫定翻訳手段は、生成したｎ個の機械語命令列のそれぞれに対応するコードサイズを前記情報として生成し、

前記決定手段は、生成されたコードサイズが最小となる機械語命令列を最適なものとして決定することを特徴とする請求項１０記載の翻訳装置。

【請求項１２】　前記暫定翻訳手段は、生成したｎ個の機械語命令列のそれぞれを前記プロセッサが実行するのに要する時間を前記情報として生成し、

前記決定手段は、生成された実行時間が最小となる機械語命令列を最適なものとして決定することを特徴とする請求項１０記載の翻訳装置。

【請求項１３】　前記翻訳装置はさらに、前記プログラムを所定の基本単位に分割する分割手段を備え、

前記暫定翻訳手段及び前記決定手段は、前記分割手段により分割された基本単位ごとに前記生成及び前記決定を行うことを特徴とする請求項１１又は１２記載の翻訳装置。

【請求項１４】　前記ｎは２であることを特徴とする請求項９又は１３記載の翻訳装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、複数の命令解読器を備えたプロセッサ及びそのコンパイラに関し、特に機械語命令列のコードサイズの縮小化を支援するプロセッサ及びコンパイラに関する。

【０００２】

【従来の技術】近年、マイクロプロセッサ（以下、単に「プロセッサ」という。）を用いた製品は、ますます高機能化が求められ、それに伴ってオブジェクトプログラム（以下、単に「オブジェクト」という。）は大規模化してきている。ところが、プロセッサ組み込み型の製品においては、オブジェクトをＲＯＭに格納して実装する必要があるため、そのコードサイズの増大は実装すべきＲＯＭの容量や個数の増大をもたらし、低価格な製品の開発を阻害する。従って、このような製品の開発においては、可能な限りコードサイズを縮小しておくことが望まれる。また、処理速度の高速化の点においても、より少ないステップ数のオブジェクトが望まれる。

【０００３】コードサイズの縮小化を支援する従来技術として、命令セットの定義において、使用頻度の高い機械語命令（以下、単に「命令」という。）を語長の短い命令に割り当てておく方法が挙げられる。具体的には、例えば、１バイト長命令と２バイト長命令の２種類の語長の命令からなる命令セットを備えるプロセッサであれば、使用頻度が高いと考えられる命令を１バイト長命令に割り当てておく方法である。

【０００４】図１７は、従来のプロセッサが備える命令のフォーマットの一例を示す図である。図１７（ａ）は１バイト長命令、図１７（ｂ）は２バイト長命令のフォーマットを示す。なお、ここでは、命令の語長とは、命令を構成するオペコードとオペランドのうちのオペコードだけの長さをいうものとする。また、オペコードは、プロセッサにフェッチされた後に命令デコーダによって解読される部分であり、その命令の種類（演算処理の種類）を特定するコードである。

【０００５】このように、命令の種類はそのオペコードの長さによって決定されるので、例えば、図１７（ａ）に示された１バイト長命令としては最大で２５６種類の命令が定義される。従って、使用頻度が高いと考えらる２５６個の命令を１バイト長命令に割り当ててプロセッサの命令デコーダを設計しておくことによって、同一処理内容を指示するオブジェクトであっても、そのような考慮をしていない場合に比べ、コードサイズを縮小することができる。

【０００６】

【発明が解決しようとする課題】しかしながら、プロセッサの高機能化が求められる今日においては、上記種類の命令だけでは十分とは言えない。特に、近年のマルチメディア技術の発展に伴い、画像データ処理関連（データ圧縮・伸長、符号・復号、積和演算等）の命令の必要性が高まっているが、従来のプロセッサでは、例えば、画像処理関連の新たな命令を従来の基本命令セットに追加するのは困難である。

【０００７】即ち、上記のような従来のプロセッサであれば、２５６種類を超える１バイト長命令を備えることはもはや不可能である。従って、従来の２５６種類の１

バイト長命令を維持したままで、新たな種類の命令を追加するためには、２バイト長命令又は２バイトを超える拡張命令を定義する必要があるが、これでは、コードサイズの縮小化が阻害される。

【０００８】そこで、本発明は、かかる問題点に鑑みてなされたものであり、一定語長の命令であっても、より多くの種類の命令を定義することができるプロセッサ及びそのための翻訳装置を提供することを目的をする。具体的には、１バイト長命令であれば、実質的に２５６種類を超える命令を備えるプロセッサ、また、２バイト長命令であれば、実質的に６５５３６種類を超える命令を備えるプロセッサ及びそのための翻訳装置を提供することを目的とする。

【０００９】

【課題を解決するための手段】上記目的を達成するために、本発明に係るプロセッサは、命令メモリから命令を読み出して解読する命令解読手段と、その命令解読手段から出力される解読結果に従って演算処理を実行する実行手段とを備えるプロセッサであって、前記命令解読手段は、前記命令メモリから命令を読み出すフェッチ手段と、前記フェッチ手段により読み出された命令を解読し、少なくとも１個の同一命令に対して異なる解読結果を出力するｎ（２以上の整数）個の命令解読器と、前記ｎ個の命令解読器から選択された一つの命令解読器の解読結果だけが前記実行手段に出力されるよう前記ｎ個の命令解読器を制御する制御手段とからなることを特徴とする。

【００１０】これによって、例えば、１バイト長命令であれば実質的に２５６種類を超える命令を備えるプロセッサ、また、２バイト長命令であれば実質的に６５５３６種類を超える命令を備えるプロセッサが実現されるので、従来の単一デコーダ型プロセッサに比べ、同一処理を行わせる機械語プログラムであってもそのコードサイズ及び実行時間は短縮される。

【００１１】また、本発明に係る翻訳装置は、プログラムを翻訳することにより、ｎ（２以上の整数）個の異なる種類の命令解読器を備えるプロセッサを対象とする機械語命令列を生成する翻訳装置であって、最適な機械語命令列を生成するためには前記ｎ個の命令解読器の中のいずれを対象とすべきかを決定する決定手段と、前記決定手段により決定された命令解読器を対象として前記プログラムを翻訳し機械語命令列を生成する生成手段とを備えることを特徴とする。

【００１２】さらに、本発明に係る翻訳装置は、プログラムを翻訳することにより、ｎ（２以上の整数）個の異なる種類の命令解読器を備えるプロセッサを対象とする機械語命令列を生成する翻訳装置であって、前記プログラムから前記ｎ個の命令解読器のそれぞれを対象とする機械語命令列とその最適化に必要な情報とを生成する暫定翻訳手段と、前記暫定翻訳手段により生成された前記

情報に基づいて前記ｎ個の機械語命令列の中から最適な１個を決定しその機械語命令列をこの翻訳装置が生成する機械語命令列とする決定手段とを備えるとすることもできる。

【００１３】これによって、複数の命令デコーダ、即ち、複数の命令セットを備えるプロセッサを対象とする翻訳装置が実現され、高級言語等による上記プロセッサの利用が可能となる。

【００１４】

【発明の実施の形態】以下、本発明に係る命令解読器切替型プロセッサ及びその翻訳装置について、図面を参照しながら具体的に説明する。

（実施の形態１）実施の形態１では、本発明に係る命令解読器切替型プロセッサについて説明する。

【００１５】図１は、本発明の実施の形態１における命令解読器切替型プロセッサの構成を示すブロック図である。本プロセッサは、大きく分けて、命令解読ブロック１００と演算ブロック１１０とからなる。

［命令解読ブロックの構成］命令解読ブロック１００は、データバス１２１を介してフェッチした命令を解読する部分であり、命令レジスタ１０１、切替命令デコーダ１０２及び８個の命令デコーダ１０３ａ～１０３ｈから構成される。なお、本図には、８個の命令デコーダ１０３ａ～１０３ｈのうち、２個の命令デコーダ（第１命令デコーダ１０３ａ及び第８命令デコーダ１０３ｈ）だけが図示され、他の６個の命令デコーダ（第２命令デコーダ１０３ｂ～第７命令デコーダ１０３ｇ）の図示は省略されている。また、各構成要素１０１～１０３ｈは、図示されていないクロック信号や制御信号の供給を受けて同期して動作する。

【００１６】命令レジスタ１０１は、１バイト長のレジスタであり、図示されていない外部メモリからデータバス１２１を介して順次にフェッチした１バイト分の命令を保持する。ここに保持された命令は、オペコードであれば切替命令デコーダ１０２と８個の命令デコーダ１０３ａ～１０３ｈの合計９個の命令デコーダに同時に送られ、一方、オペランドであれば演算ブロック１１０に転送される。なお、説明の容易のため、本プロセッサは、１種類の命令フォーマット、具体的には、図１７（ａ）に示された１バイト長命令のみを備えるものとする。

【００１７】切替命令デコーダ１０２は、８個の命令デコーダ１０３ａ～１０３ｈのうちの１個を有効とさせるための制御回路であり、図２に示されるように、さらに、切替命令専用解読器１０２ａ及びラッチ１０２ｂから構成される。切替命令専用解読器１０２ａは、比較器とデコーダ等からなり、命令レジスタ１０１から送られてきた命令が切替命令であるかどうかを判断し、切替命令である場合には、その命令の一部をデコードすることにより、８個の命令デコーダ１０３ａ～１０３ｈのうちの１個のみを有効化させる８ビットのデータパターンをラッチ１０２ｂに保持させる。

【００１８】ここで、切替命令とは、図３に示される８種類の命令のいずれかをいい、切替命令であることを示す５ビットと８個の命令デコーダ１０３ａ～１０３ｈのうちの１個を特定する３ビットからなる。８個の命令デコーダ１０３ａ～１０３ｈは、命令レジスタ１０１から送られてきた１バイト長命令を同時に解読し、それらの解読結果は、切替命令デコーダ１０２のラッチ１０２ｂからの信号により、いずれか１つのみが選択されて制御信号線１０４に出力される。

【００１９】各命令デコーダ１０３ａ～１０３ｈは、いずれも、図４に示されるように、１バイト幅を基本とするプリデコーダ４０１、シーケンサ４０２、マイクロＲＯＭ４０３及び出力バッファ４０４から構成され、形式的な構成はそれぞれ同一である。プリデコーダ４０１は、入力された１バイト長のオペコードを、対応するマイクロコードが格納されたマイクロＲＯＭ４０３のエントリアドレスに変換するものである。シーケンサ４０２は、プリデコーダ４０１が示すエントリアドレスから始まるマイクロコードをマイクロＲＯＭ４０３から出力させるためのアドレス発生用カウンタである。マイクロＲＯＭ４０３は、各命令に対応するマイクロコードを記憶している。出力バッファ４０４は、ゲート回路であり、切替命令デコーダ１０２のラッチ１０２ｂからのゲート信号による制御の下に、マイクロＲＯＭ４０３から出力されたマイクロコードを制御信号線１０４に出力したり、禁止したり（出力をハイインピーダンスに）する。

【００２０】ここで、８個の命令デコーダ１０３ａ～１０３ｈ相互の関係は、以下の通りである。８個の命令デコーダ１０３ａ～１０３ｈは、いずれも２５６種類の１バイト長命令が定義された命令セットを有しているが、それら命令セットは、完全には一致しない。具体的には、各命令デコーダ１０３ａ～１０３ｈが備えるマイクロＲＯＭ４０３は、２５６種類の命令のうちの一部（ここでは、３２種類）の命令に対するマイクロコードが異なり、他の多く（ここでは、２２４個）の命令に対するマイクロコードが一致する。そのために、同一のオペコードを有する命令を解読した場合であっても、各命令デコーダ１０３ａ～１０３ｈは、異なる解読結果を出力することがある。即ち、同一のオペコードを有する命令が本プロセッサにフェッチされた場合であっても、その時点で有効とされている命令デコーダの種類によっては制御信号線１０４に出力される信号が異なり、演算ブロック１１０での動作が異なることがある。

【００２１】ここで、上記３２種類の命令、即ち、オペコードが同一であるが解読される命令デコーダに依存してプロセッサの動作が異なるようなものを「デコーダ依存型命令」、一方、残る２２４種類の命令、即ち、解読される命令デコーダに依存することなくプロセッサの動作が同一となるものを「デコーダ非依存型命令」と呼

ぶ。なお、デコーダ依存型命令の数をデコーダ非依存型命令よりも少なくしているのは、デコーダ依存型命令の数を多くした場合に生ずるマイナス効果、即ち、有効とする命令デコーダを頻繁に切り替えなければならないことから生ずるオーバーヘッド（処理速度の低下）を回避するためである。

【００２２】図５は、本プロセッサの命令セット、即ち、上記８個の命令デコーダ１０３ａ～１０３ｈを一体とした場合の命令セットの一部を示すテーブルである。なお、本図で用いられている、”opn”はオペランド、”label”は分岐先を示すラベルを意味する。図５に示された命令の中では、”ｓｐ１”だけがデコーダ依存型命令であり、解読される命令デコーダの種類によってプロセッサの動作が異なる。なお、本図には、これら異なる８種類の動作のうち２種類だけが示されている。ここで、ゼロラン長とは、ビット並びにおいて初めて「１」が出現する場合のその位置をいい、例えば、ビット並び「００１１０１００」であれば最下位ビット（最右端）からのゼロラン長は３である。この命令は、画像認識処理等において必要とされる命令の一つである。

［演算ブロックの構成］演算ブロック１１０は、命令解読ブロック１００での解読結果に従って演算やデータ転送等の処理を実行する部分であり、データセレクタ１１１、レジスタファイル１１２、プログラムカウンタ１１３、ＡＬＵ１１４、インクリメンタ１１５、アドレスバッファ１１６及びデータバッファ１１７から構成される。

【００２３】この演算ブロック１１０は、通常のプロセッサが備えるものと異ならない。各構成要素１１１～１１７は、命令解読ブロック１００からの制御信号線１０４に接続され、その信号によって演算やデータ転送等の処理を行う。データセレクタ１１１は、内部Ａバス１１８か又はデータバッファ１１７からのデータを選択してレジスタファイル１１２に送る。レジスタファイル１１２には、スタックポインタや１６個の汎用レジスタ等が含まれる。プログラムカウンタ１１３は、命令解読ブロック１００にフェッチされた命令が置かれていた外部メモリ上のアドレス等を保持する。プログラムカウンタ１１３に保持されたアドレスは、インクリメンタ１１５によって１だけ加算され、次にフェッチすべき命令のアドレスとしてアドレスバッファ１１６を介してアドレスバス１２０に出力されたり、プログラムカウンタ１１３に送られたりする。ＡＬＵ１１４は、内部Ａバス１１８及び内部Ｂバス１１９のそれぞれから送られてくるデータの算術演算を行う。内部Ｂバス１１９は、主に、命令レジスタ１０１に保持されたオペランドを演算ブロック１１０に転送するのに用いられる。

［命令解読器切替型プロセッサの動作］以上のように構成された本プロセッサの動作について、具体的な命令を実行する場合を例にとって説明する。

【００２４】図６は、実行の対象となるオブジェクトのリストである。このリストで用いられている個々の命令の意味は、図５に示されるテーブルの通りである。また、オペランドに用いられているアドレシングモードの意味は、以下の通りである。

ａ　：　変数ａの値

＆ｂ　：　変数ｂが格納されている外部メモリ上のアドレス

（ｃ）　：　外部メモリ上のアドレスｃに格納されている値

図６に示されたオブジェクトは、大きく分けて、ラベル＿ｍ、＿ｆ及び＿ｇで始まる３つのブロックからなる。

【００２５】ラベル＿ｍから始まるブロック（ステップＳ６００～Ｓ６０４）は、ｃ言語でいうメイン関数に相当し、引き数ｔ１を設定した後に関数ｆを呼び出し、続いて、再び引き数ｔ１を設定した後に関数ｇを呼び出すものである。なお、関数ｆでの処理と関数ｇでの処理は特に関連するものではない。ラベル＿ｆから始まるブロック（ステップＳ６０５～Ｓ６０７）は、関数ｆに相当し、データブロックのコピーを行うものである。

【００２６】ラベル＿ｇから始まるブロック（ステップＳ６０８～Ｓ４１６）は、関数ｇに相当し、データブロックの各要素についてのゼロラン長を比較するものである。このようなオブジェクトを実行した場合の本プロセッサの具体的な動作は、以下の通りである。なお、このオブジェクトの実行に先立ち、切替命令デコーダ１０２は初期化され、第１命令デコーダ１０３ａが有効とされているものとする。

［ステップＳ６００、Ｓ６０１］プロセッサは、第１命令デコーダ１０３ａを用いて、関数ｍの先頭から実行を開始し関数ｆを呼び出すが、これらのステップにおいては、結果的には１個の命令デコーダのみからなる従来のプロセッサ（以下、「単一デコーダ型プロセッサ」と呼ぶ。）の場合と同一の動作をする。

［ステップＳ６０５］命令”ｅｘ８”は、切替命令デコーダ１０２によって解読され、その結果、ラッチ１０２ｂに保持されるパターンは、第８命令デコーダ１０３ｈを有効とするパターンに更新される。これによって、次の命令（ステップＳ６０６）からは、第８命令デコーダ１０３ｈによって解読されることになる。

【００２７】なお、切替命令は、他の命令と同様に、切替命令デコーダ１０２に解読されると同時に、各命令デコーダ１０３ａ～１０３ｈによっても解読されるが、いずれの命令デコーダも、切替命令に対してはノーオペレーション動作となるよう設計されている。

［ステップＳ６０６］命令”ｓｐ１”は、第８命令デコーダ１０３ｈによって解読されるので、その結果、図５に示されるように、アドレスｂから始まる外部メモリ上の１０個のデータがアドレスｔ１から始まる領域にコピーされる。

［ステップＳ６０７、Ｓ６０２、Ｓ６０３］次に、プロセッサは、関数ｆでの処理を終えて関数ｍに戻った後に関数ｇを呼び出すが、これらのステップにおいても、結果的には従来の単一デコーダ型プロセッサの場合と同一の動作をする。

［ステップＳ６０８］命令”ｅｘ１”は、切替命令デコーダ１０２によって解読され、その結果、ラッチ１０２ｂに保持されるパターンは、第１命令デコーダ１０３ａを有効とするパターンに更新される。これによって、次の命令（ステップＳ６０９）からは、再び第１命令デコーダ１０３ａによって解読されることになる。

［ステップＳ６０９～Ｓ６１６、Ｓ６０４］ステップＳ６１０においては、再び命令”ｓｐ１”が実行されるが、ステップＳ６０６の場合と相違し、ここでは第１命令デコーダ１０３ａによって解読されるので、図５に示されるように、アドレスａからｉ番目に位置する外部メモリ上のデータのゼロラン長が算出され変数ｊに書き込まれる。

【００２８】ステップＳ６１１における動作もステップＳ６１０の場合と同様であり、アドレスｂからｉ番目に位置する外部メモリ上のデータのゼロラン長が算出され変数ｋに書き込まれる。このように、第８命令デコーダ１０３ｈが有効とされていたステップＳ６０６と第１命令デコーダ１０３ａが有効とされていたステップＳ６１０及びＳ６１１とにおける動作を比較して判るように、本プロセッサは、同一オペコードの命令”ｓｐ１”に対して、２種類の異なる動作を行った。

【００２９】なお、ステップＳ６０９～Ｓ６１５においては、アドレスａから始まる１０個のデータとアドレスｂから始まる１０個のデータのそれぞれについてゼロラン長を比較し、不一致が発見された場合には、図示されていない関数errorを呼び出す処理が行われる。以上のように、本プロセッサは、１バイト長のオペコードで定義される命令セット、即ち、形式的に２５６種類の命令を備えるが、それら命令の一部はデコーダ依存型命令であることから、実質的に２５６種類を超える命令を備えることに等しい。

【００３０】具体的には、本プロセッサは、２２４個のデコーダ非依存型命令と３２個のデコーダ依存型命令を有するので、１個のデコーダ依存型命令が実質的に８個の命令に相当することから、合計４８０（＝２２４＋３２×８）個の命令を備えることに等しい。即ち、本プロセッサの如く複数の命令解読デコーダを備えることによって、１バイト長命令のフォーマットを維持したままで、従来の２５６種類の命令からなる命令セットに加えて、さらに、２２４（＝４８０－２５６）種類の新たな命令を定義することが可能となる。

【００３１】また、本プロセッサを構成する各命令デコーダは、相互に形式的な構成が同一であり、しかも、従来のプロセッサの命令デコーダと同様に、８ビット幅の命令を解読するものである。従って、従来のプロセッサが有する解読器をそのまま流用する形で、大幅に設計変更することなく、本プロセッサを容易に構築することができる。

【００３２】さらに、本プロセッサにおいては、切替命令デコーダ１０２と他の８個の命令デコーダ１０３ａ～１０３ｈとは、その処理内容が異なる独立した回路であるため、パイプライン構成とすることも容易である。なお、本実施形態においては、プロセッサは、８個の命令デコーダを備え、１種類の１バイト長命令だけを実行したが、本発明は、これら数値に限定されるものではない。例えば、プロセッサは、２個の命令デコーダを備え、２バイト長命令を実行し、各命令デコーダは、それぞれ３２種類のデコーダ依存型命令と６５５０４種類のデコーダ非依存型命令を備えるとすることもできる。これによって、２バイト長命令のフォーマットを維持したままで、従来の６５５３６種類の命令からなる命令セットに加えて、さらに、３２（＝３２×２＋６５５０４－６５５３６）種類の新たな命令を定義することが可能となる。

【００３３】また、本実施形態においては、８個の命令デコーダのそれぞれのマイクロＲＯＭは、共通する部分（２５６個のうちの２２４個の命令に相当する部分）を冗長に有しているが、このような構成に限られるものではない。例えば、８個の命令デコーダは、相互に異なる部分（２５６個のうちの３２個の命令に相当する部分）だけをマイクロＲＯＭに有し、共通する部分は、これら８個の命令デコーダとは異なる別の１個の命令デコーダのマイクロＲＯＭに格納されるとすることもできる。即ち、デコーダ依存型命令だけが８個の命令デコーダのいずれかによって解読され、一方、デコーダ非依存型命令は共通の１個の命令デコーダによって解読されることとなる。これによって、命令デコーダの切替頻度は多少増えるものの、必要とされるハードウェア回路を削減することができる。

（実施の形態２）実施の形態２では、本発明に係る命令解読器切替型プロセッサの他の実現態様について説明する。

【００３４】本実施形態におけるプロセッサは、実施形態１におけるプロセッサと同一の機能、即ち、同一の命令セットを備えるが、その実現方法が異なる。具体的には、本実施形態におけるプロセッサは、大きく分けて命令解読ブロックと演算ブロックの２個のブロックからなる点において、実施形態１のプロセッサと同じであるが、命令解読ブロックの内部構成が異なる。以下、実施形態１と異なる点を中心に説明する。

［命令解読ブロックの構成］図７は、本発明の実施の形態２における命令解読器切替型プロセッサの命令解読ブロック７００のみの構成を示すブロック図である。

【００３５】命令解読ブロック７００は、命令レジスタ

７０１、デマルチプレクサ７０２及び８個の命令デコーダ７０３ａ～７０３ｈから構成され、命令レジスタ７０１から出力された１バイト長の命令は、デマルチプレクサ７０２によって８個の命令デコーダ７０３ａ～７０３ｈのいずれか１個にのみ振り分けられる点で、実施形態１と異なる。

【００３６】デマルチプレクサ７０２は、図８に示されるように、さらに、ラッチ７０２ａ及び切替器７０２ｂからなる。ラッチ７０２ａは、現時点で有効とする命令デコーダを特定する情報、即ち、８個の命令デコーダ７０３ａ～７０３ｈのいずれか１個から出力された切替信号を保持し、切替器７０２ｂを制御する。切替器７０２ｂは、ラッチ７０２ａからの制御の下に、命令レジスタ７０１からの１バイト長命令を８個の命令デコーダ７０３ａ～７０３ｈのいずれか１個に通過させ、他の７個の命令デコーダに対してはデータ”ｆｆ（ｈｅｘ）”を出力する。

【００３７】８個の命令デコーダ７０３ａ～７０３ｈは、いずれも、図９に示されるように、さらに、１バイト幅を基本とするプリデコーダ９０１、シーケンサ９０２、マイクロＲＯＭ９０３及び出力バッファ９０４から構成され、形式的な構成はそれぞれ同一である。これら命令デコーダと実施形態１のものとの主な相違点は以下の２点である。

【００３８】即ち、第１に、プリデコーダ９０１は、内臓する比較器により、データ”ｆｆ（ｈｅｘ）”が入力されたときには、出力バッファ９０４からの出力を禁止する。これは、自己の命令デコーダが選択されていないと判断した場合に相当する。一方、”ｆｆ（ｈｅｘ）”を除くデータが入力されたときには、プリデコーダ９０１は、実施形態１の場合と同様の動作を行う。これは、自己の命令デコーダが有効であると選択されている場合に相当する。

【００３９】第２に、８個の命令デコーダ７０３ａ～７０３ｈが備える全てのマイクロＲＯＭ９０３は、８種類の切替命令に対応するマイクロコードを有している。以上のように構成された命令解読ブロック７００により、実施形態１の場合と異なり、命令レジスタ７０１に保持された命令は、８個の命令デコーダ全てによって解読されるのではなく、有効とされている１個の命令デコーダにのみ送られて解読されることになる。これによって、有効とされていない７個の命令デコーダでの無駄な解読動作が回避され、また、切替命令だけを解読する専用のデコーダが不要となる。

【００４０】なお、実施形態１及び２においては、各命令デコーダ自体は、マイクロプログラム方式により実現されたが、この方式に代えて、ＲＩＳＣ（Reduced Instruction Set Computer）等に採用されているワイヤードロジック方式により実現されてもよい。即ち、本発明は、命令デコーダ自体の実現方法を特徴とするものではなく、第１に、複数の命令デコーダ、即ち、複数の命令セットを備えることを特徴とし、第２に、最後に解読・実行した切替命令の種類、即ち、その時点におけるプロセッサの状態に依存して、それら複数の命令セットの中から使用すべき１個の命令セットが決定されることを特徴とするものである。

（実施の形態３）次に、本発明に係る翻訳装置の一例、ここでは、上記実施形態１のプロセッサをターゲットとする翻訳装置について説明する。

【００４１】図１０は、本実施形態における翻訳装置の構成を示す機能ブロック図である。本装置は、汎用のコンピュータシステム及びその上で実行されるプログラムによって実現されるものであり、機能的には、ブロック分割部１００１、中間コード生成部１００２、コード生成部１００３、オブジェクト決定部１００４、パラメータ獲得部１００５及び命令テーブル１００６から構成される。なお、図１０には、これら各構成要素１００１～１００６の処理対象となる入出力データ１０１０～１０２２も同時に示されている。

【００４２】パラメータ獲得部１００５は、操作者からの指示に従って、本装置が有する選択的な機能（オプション）を指定する情報を獲得し、その情報をブロック分割部１００１及びオブジェクト決定部１００４に通知する。オプションには分割パラメータと最適化パラメータの２種類がある。「分割パラメータ」は、使用される命令デコーダ（命令セット）の種類が切り替えられることがない最小のプログラム単位（以下、「ブロック」という。）の種類（例えば、「関数」、途中に分岐の入口出口を持たない「基本ブロック」等）を指定するものであり、一方、「最適化パラメータ」は、オブジェクトを最適化する際の基準（例えば、「コードサイズ」、「クロックサイクル数」等）を指定するものである。

【００４３】ブロック分割部１００１は、パラメータ獲得部１００５から通知される分割パラメータに基づいて、ｃ言語等で書かれたソースプログラム１０１０を複数のブロック１０１１～１０１３に分割する。中間コード生成部１００２は、分割された各ブロック１０１１～１０１３ごとに、字句・構文解析をした後に３オペランド表現による中間コードを生成することにより、各ブロックに対応する中間ファイル１０１４～１０１６を出力する。

【００４４】命令テーブル１００６は、図１１に示される情報を予め保持しているものであり、中間コード生成部１００２が使用し得る全ての種類の中間コード１１０１と、各中間コードに対応する８組の情報１１０２とからなる。この８組の情報１１０２は、８個の命令デコーダ１０３ａ～１０３ｈのそれぞれをターゲットとした場合における各中間コードに対応する命令、その命令のサイズ（バイト数）及びその命令の解読・実行に要するクロックサイクル数からなる。

【００４５】コード生成部１００３は、中間コード生成部１００２によって生成された各中間ファイル１０１４～１０１６ごとに、命令テーブル１００６を参照することにより、各命令デコーダ１０３ａ～１０３ｈに対応する８種類のオブジェクト１０１７ａ～１０１７ｈ、１０１８ａ～１０１８ｈ、１０１９ａ～１０１９ｈを生成する。このとき、コード生成部１００３は、命令テーブル１００６を参照することによって、各中間コードを命令に変換するだけでなく、各中間コードに対応する全てのサイズ及びクロックサイクル数の合計値をも算出して各オブジェクトに書き込んでおく。

【００４６】オブジェクト決定部１００４は、ブロックごとに、コード生成部１００３によって生成された８種類のオブジェクトから最適な１種類を決定し、決定したそのオブジェクトをそのブロックの最終オブジェクトとして出力する。このとき、オブジェクト決定部１００４は、パラメータ獲得部１００５から通知される最適パラメータに基づき、合計サイズ又は合計クロックサイクル数のいずれかが最小となるオブジェクトを最適なものとして決定する。

【００４７】以上のように構成された本発明に係る翻訳装置の動作について、具体的なソースプログラムが入力された場合を例にとって説明する。図１２は、本装置の動作手順を示すフローチャートである。図１３は、ｃ言語で書かれたソースプログラム１０１０のリストである。なお、このソースプログラム１０１０は、実施形態１における図６に示されたアセンブラプログラムと同一の処理、即ち、メイン関数mからブロックコピーを行う関数ｆとゼロラン長の比較を行う関数ｇを順次に呼び出すものである。

【００４８】まず、パラメータ獲得部１００５は、操作者から分割パラメータ（ここでは、「関数」）と最適パラメータ（ここでは、「コードサイズ」）の指定を受ける（ステップＳ１２０１）。ブロック分割部１００１は、パラメータ獲得部１００５から通知される分割パラメータ（「関数」）に基づいて、図１３に示されたソースプログラム１０１０を３個の関数m，ｆ，ｇに分割する（ステップＳ１２０２）。

【００４９】そして、分割された各関数m，ｆ，ｇごとに、以下の処理が繰り返される（ステップＳ１２０３～Ｓ１２０９）。中間コード生成部１００２は、各関数m，ｆ，ｇごとに、中間コードを生成することにより、３個の中間ファイルm，ｆ，ｇを出力する（ステップＳ１２０４）。

【００５０】続いて、コード生成部１００３は、各中間ファイルm，ｆ，ｇごとに、命令テーブル１００６を参照することにより、８種類のオブジェクトm1～m8、ｆ1～ｆ8、ｇ1～ｇ8を生成する（ステップＳ１２０５～Ｓ１２０７）。このとき、コード生成部１００３は、命令テーブル１００６を参照することによって、命令だけでなく、各中間コードに対応する全ての命令のサイズ及びクロックサイクル数の合計値をも算出して各オブジェクトに書き込んでおく（ステップＳ１２０６）。

【００５１】図１４は、オブジェクトm1～m8のそれぞれのリストである。８個のオブジェクトm1～m8は全く同一であるので、図１４にはそのうちの一つが示されている。これは、中間ファイルmには命令デコーダに依存する中間コードが存在しなかったことによる。なお、このリストの末尾２行は、「このオブジェクトの合計サイズは１１であり、合計クロックサイクル数は１６である。」ことを示してい。

【００５２】図１５（ａ）は、オブジェクトｆ1、即ち、第１命令デコーダ１０３ａをターゲットとする関数ｆのオブジェクトのリストである。ここでは、ブロック転送の処理がデコーダ非依存型命令”ｍｏｖ”等により実現されていることが分かる。図１５（ｂ）は、オブジェクトｆ8、即ち、第８命令デコーダ１０３ｈをターゲットとする関数ｆのオブジェクトのリストである。ここでは、ブロック転送の処理がデコーダ依存型命令”ｓｐ１”により実現されていることが分かる。なお、他の６個のオブジェクトｆ2～ｆ7のリストは省略されている。

【００５３】同様に、図１６（ａ）は、オブジェクトｇ1、即ち、第１命令デコーダ１０３ａをターゲットとする関数ｇのオブジェクトのリストである。ここでは、ゼロラン長比較の処理がデコーダ依存型命令”ｓｐ１”により実現されていることが分かる。図１６（ｂ）は、オブジェクトｇ8、即ち、第８命令デコーダ１０３ｈをターゲットとする関数ｇのオブジェクトのリストである。ここでは、ゼロラン長比較の処理がデコーダ非依存型命令”ｍｏｖ”等により実現されていることが分かる。

【００５４】最後に、オブジェクト決定部１００４は、パラメータ獲得部１００５から通知される最適パラメータ（サイズ）に基づき、関数m，ｆ，ｇごとに、コード生成部１００３によって生成された８種類のオブジェクトの合計サイズを読み出して比較することにより、その合計サイズが最小となるオブジェクトを１個だけ選択する（ステップＳ１２０８）。具体的は、関数mについては、合計サイズに差がないために図１４に示されるオブジェクトm1が選択されるが、関数ｆについては、合計サイズが最小である図１５（ｂ）に示されたオブジェクトｆ8が選択され、関数ｇについては、合計サイズが最小である図１６（ａ）に示されたオブジェクトｇ1が選択される。

【００５５】このようにして、全ての関数m，ｆ，ｇのそれぞれについて選択された３個のオブジェクトm1，ｆ8，ｇ1は、オブジェクト決定部１００４によって１個に結合された後、図６に示される最終オブジェクトとして出力される（ステップＳ１２１０）。以上ようにして、本装置に入力されたソースプログラム１０１０は、コードサイズを最小にする観点より、関数ごとに、ター

ゲットとすべき命令デコーダが決定され、その決定に基づくオブジェクトが生成された。これにより、従来の単一デコーダ型プロセッサをターゲットとする翻訳装置に比べ、より小さいブロック単位でコードサイズの最適化が行われるため、従来の翻訳装置によって出力されるオブジェクト（例えば、オブジェクトｍｌ＋オブジェクトｆｌ＋オブジェクトｇｌ）よりもコードサイズの小さいオブジェクトが生成される。

【００５６】なお、本実施形態では、分割パラメータが「関数」でかつ最適パラメータが「コードサイズ」と指定されたが、本発明は、このような場合に限定されるものでない。例えば、分割パラメータが「基本ブロック」でかつ最適パラメータが「クロックサイクル数」と指定された場合には、ブロック分割部１００１はソースプログラム１０１０を基本ブロックに分割し、オブジェクト決定部１００４は合計サイクル数が最小のオブジェクトを決定するので、これにより、実行時間が最小となるオブジェクトの組合せが最終オブジェクトとして出力されることになる。

【００５７】また、最適なオブジェクトを決定する要因として１種類のパラメータに限られることはなく、例えば、命令サイズとクロックサイクル数の合計値を決定要因とすることもできる。さらに、本実施形態では、一旦、８種類のオブジェクトを生成した後に（ステップＳ１２０６）それらの中から最適なものを選択したが（ステップＳ１２０８）、その順序に限られるものではなく、逆の順序であってもよい。

【００５８】具体的には、１個の中間ファイルに対して、各中間コードごとに命令テーブルを参照する操作を２回繰り返す。即ち、１回目の参照においては、命令を参照することなく命令サイズだけを参照することによって８種類の命令セットの中から最小サイズとなり得る１個を決定し、続いて、２回目の参照においては、いま決定した命令セットだけを用いて各中間コードに対応する命令だけを参照することによって１個の最終オブジェクトを生成する。これによって、本実施形態の場合と比較し、命令テーブルを参照する操作が１回ではなく２回に増加したものの、８個の暫定的なオブジェクトを生成するステップが回避されることから、翻訳に必要な作業領域（一時的なメモリ領域）の容量が削減される。

【００５９】また、本実施形態では、８種類の命令セットは、１個の命令テーブル１００６に格納されていたが、８個の命令テーブルに分割されて格納される構成であってもよい。そして、８種類の命令テーブルのそれぞれに、中間コードから命令を生成するための他の必要な情報（例えば、変数の資源への割り付け情報等）を含ませることにより、これらの情報を加味した命令生成が可能となる。

【００６０】

【発明の効果】以上の説明から明らかなように、本発明

に係るプロセッサは、命令メモリから命令を読み出して解読する命令解読手段と、その命令解読手段から出力される解読結果に従って演算処理を実行する実行手段とを備えるプロセッサであって、前記命令解読手段は、前記命令メモリから命令を読み出すフェッチ手段と、前記フェッチ手段により読み出された命令を解読し、少なくとも１個の同一命令に対して異なる解読結果を出力するｎ（２以上の整数）個の命令解読器と、前記ｎ個の命令解読器から選択された一つの命令解読器の解読結果だけが前記実行手段に出力されるよう前記ｎ個の命令解読器を制御する制御手段とからなることを特徴とする。

【００６１】これによって、例えば、１バイト長命令であれば実質的に２５６種類を超える命令を備えるプロセッサ、また、２バイト長命令であれば実質的に６５５３６種類を超える命令を備えるプロセッサが実現されるので、従来の単一デコーダ型プロセッサに比べ、同一処理を行わせる機械語プログラムであってもコードサイズ及び実行時間の短縮化が図られる。

【００６２】ここで、前記制御手段は、前記フェッチ手段により読み出された命令が所定の命令解読器切替命令であるかどうかを判断する判断部と、前記判断部により所定の命令解読器切替命令であると判断された場合には、その命令解読器切替命令に基づいて前記選択を行う選択部とからなるとすることもできる。これによって、命令解読器は命令によって切り替えられるので、プログラム中に命令解読器切替命令を配置しておくことにより、複数の命令解読器の中から所望の命令解読器を所望のタイミングで有効とさせることができる。

【００６３】さらに、前記選択部は、前記判断部により所定の命令解読器切替命令であると判断された場合には、その命令解読器切替命令から一の命令解読器を特定する情報を抽出する抽出部と、前記抽出部により抽出された情報を記憶する記憶部と、前記ｎ個の命令解読器のうち前記記憶部に記憶された情報に対応する命令解読器の解読結果だけが前記実行手段に出力されるよう前記ｎ個の命令解読器を制御する出力制御部とからなるとすることもできる。

【００６４】そして、前記出力制御部はさらに、前記ｎ個の命令解読器の解読結果が前記実行手段に出力されることを命令解読器ごとに許可又は禁止するゲート部と、前記記憶部に記憶された情報に基づいて前記ゲート部を制御するゲート制御部とからなるとすることもできる。これによって、選択された命令解読器は次にいずれかの命令解読器が選択されるまで有効とされ、また、有効とされた１個の命令解読器の解読結果だけがゲート部を通過して出力されるので、現在のプロセッサの状態に応じて複数の命令セットの中から有効とすべき１個の命令セットを決定するという状態遷移を容易に実現することができる。

【００６５】ここで、上記命令解読器の数ｎを２とする

ことともできる。これによって、有効な命令解読器を特定するための情報や命令解読器を有効とさせるための制御信号は１ビットで足りるので、回路規模の増加を最小限に抑制しつつ上記プロセッサを実現することが可能となる。本発明に係る翻訳装置は、プログラムを翻訳することにより、ｎ（２以上の整数）個の異なる種類の命令解読器を備えるプロセッサを対象とする機械語命令列を生成する翻訳装置であって、最適な機械語命令列を生成するためには前記ｎ個の命令解読器の中のいずれを対象とすべきかを決定する決定手段と、前記決定手段により決定された命令解読器を対象として前記プログラムを翻訳し機械語命令列を生成する生成手段とを備えることを特徴とする。

【００６６】これによって、複数の命令デコーダ、即ち、複数の命令セットを備えるプロセッサを対象とする翻訳装置が実現され、高級言語等による上記プロセッサの利用が可能となる。また、前記決定手段によって最適な命令解読器が決定された後にその１個の命令解読器だけを対象とする機械語命令列が生成されるので、複数の機械語命令列を生成する方法に比べ、翻訳に必要な作業用メモリの容量が削減される。

【００６７】ここで、前記決定手段は、生成された機械語命令列のコードサイズが最小となる場合やその機械語命令列を前記プロセッサが実行するのに要する時間が最小となる場合を最適として前記決定を行うこともできる。これによって、従来の単一デコーダ型プロセッサを対象とする翻訳装置に比べ、同一処理を行わせるソースプログラムであっても、コードサイズや実行時間がより短縮化された機械語命令列が生成される。

【００６８】また、前記翻訳装置はさらに、前記プログラムを所定の基本単位に分割する分割手段を備え、前記決定手段及び前記生成手段は、前記分割手段により分割された基本単位ごとに前記決定及び前記生成を行うとすることもできる。これによって、基本単位よりも小さい処理単位で命令解読器を頻繁に切り替えることから生じる処理速度の低下という不具合の発生が回避される。

【００６９】また、本発明に係る翻訳装置は、プログラムを翻訳することにより、ｎ（２以上の整数）個の異なる種類の命令解読器を備えるプロセッサを対象とする機械語命令列を生成する翻訳装置であって、前記プログラムから前記ｎ個の命令解読器のそれぞれを対象とする機械語命令列とその最適化に必要な情報とを生成する暫定翻訳手段と、前記暫定翻訳手段により生成された前記情報に基づいて前記ｎ個の機械語命令列の中から最適な１個を決定しその機械語命令列をこの翻訳装置が生成する機械語命令列とする決定手段とを備えるとすることもできる。

【００７０】これによって、ｎ個の機械語命令列と共に各機械語命令列の最適化に必要な情報が生成されるので、例えば、中間コードに対応する機械語命令のテーブル検索と中間コードに対応する最適化情報のテーブル検索という２種類のテーブル検索が１種類のテーブル検索だけで終了し、翻訳に要する時間が短縮されるという効果がある。

【図面の簡単な説明】

【図１】本発明の実施の形態１における命令解読器切替型プロセッサの構成を示すブロック図である。

【図２】同プロセッサの切替命令デコーダ１０２の詳細な構成を示すブロック図である。

【図３】同プロセッサが備える切替命令の種類を示す図である。

【図４】同プロセッサの命令デコーダ１０３ａ～１０３ｈの詳細な構成を示すブロック図である。

【図５】同プロセッサが備える命令セットの一部を示す図である。

【図６】同プロセッサが実行する対象となるオブジェクトのリストを示す図である。

【図７】本発明の実施の形態２における命令解読器切替型プロセッサの命令解読ブロック７００の構成を示すブロック図である。

【図８】同プロセッサのデマルチプレクサ７０２の詳細な構成を示すブロック図である。

【図９】同プロセッサの命令デコーダ７０３ａ～７０３ｈの詳細な構成を示すブロック図である。

【図１０】本発明の実施の形態３における翻訳装置の構成を示す機能ブロック図である。

【図１１】同翻訳装置の命令テーブル１００６に保持されている情報の一覧を示す図である。

【図１２】同翻訳装置の動作手順を示すフローチャートである。

【図１３】同翻訳装置に入力されるソースプログラム１０１０のリストを示す図である。

【図１４】図１０に示されたオブジェクトｍ1～ｍ8の一つのリストを示す図である。

【図１５】図１５（ａ）は、図１０に示されたオブジェクトｆ1のリストを示す図である。図１５（ｂ）は、図１０に示されたオブジェクトｆ8のリストを示す図である。

【図１６】図１６（ａ）は、図１０に示されたオブジェクトｇ1のリストを示す図である。図１６（ｂ）は、図１０に示されたオブジェクトｇ8のリストを示す図である。

【図１７】図１７（ａ）は、従来のプロセッサが備える１バイト長命令の命令フォーマットを示す図である。図１７（ｂ）は、従来のプロセッサが備える２バイト長命令の命令フォーマットを示す図である。

【符号の説明】

１００　命令解読ブロック
１０１　命令レジスタ
１０２　切替命令デコーダ

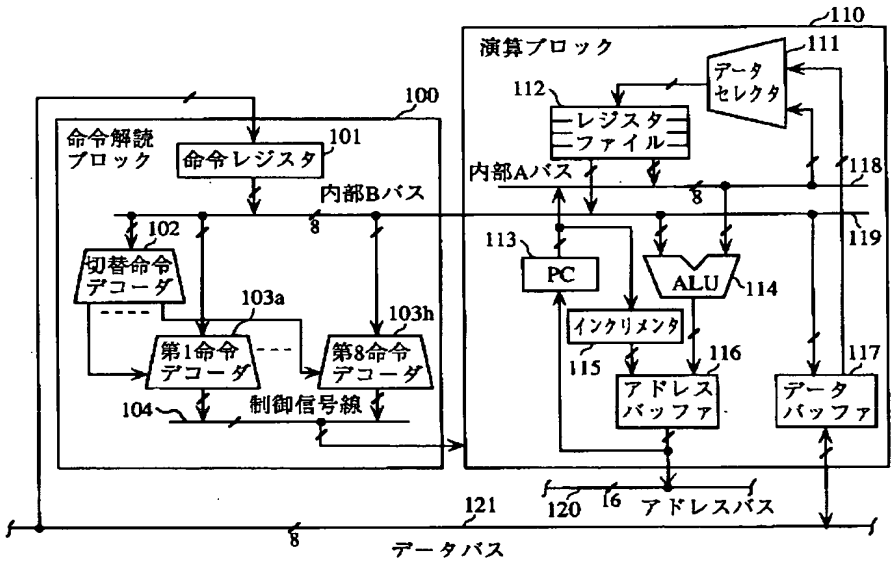
19

１０２ａ　切替命令専用解読器
１０２ｂ　ラッチ
１０３ａ～１０３ｈ　命令デコーダ
１０４　制御信号線
１１０　演算ブロック
４０１　プリデコーダ
４０２　シーケンサ
４０３　マイクロＲＯＭ

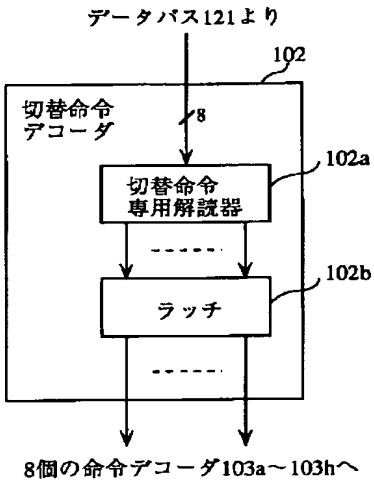
20

４０４　出力バッファ
１００１　ブロック分割部
１００２　中間コード生成部
１００３　コード生成部
１００４　オブジェクト決定部
１００５　パラメータ獲得部
１００６　命令テーブル

【図１】



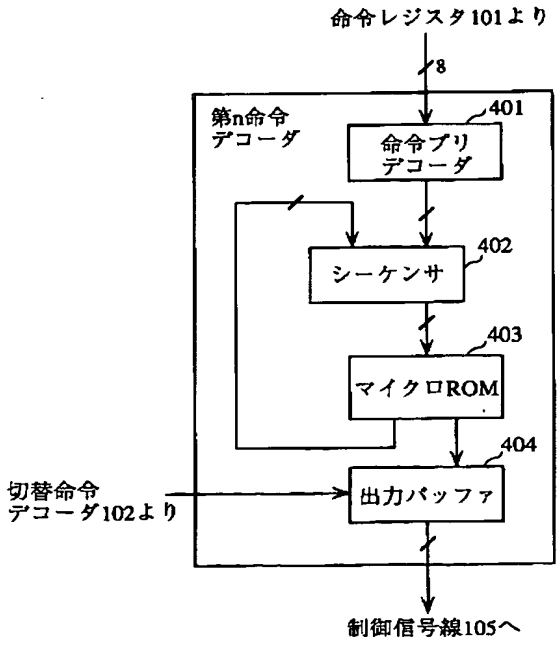
【図２】



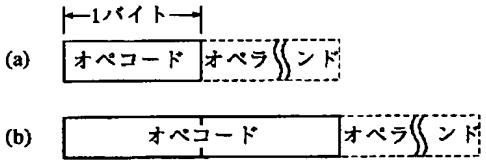
【図３】

命令	動作内容
ex1	第1命令デコーダ103aを有効にさせる
ex2	第2命令デコーダ103bを有効にさせる
ex3	第3命令デコーダ103cを有効にさせる
ex4	第4命令デコーダ103dを有効にさせる
ex5	第5命令デコーダ103eを有効にさせる
ex6	第6命令デコーダ103fを有効にさせる
ex7	第7命令デコーダ103gを有効にさせる
ex8	第8命令デコーダ103hを有効にさせる

【図４】



【図１７】



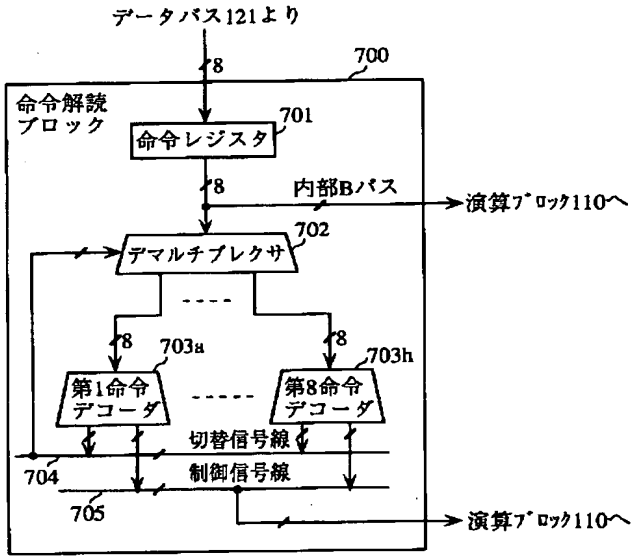
【図５】

命令	動作内容
add op1,op2	op1をop2に加算
and op1,op2	op1とop2の論理積をとり結果をop2に書き込む
beq op1,op2,label	op1＝op2ならばlabelへ分岐
blt op1,op2,label	op1＜op2ならばlabelへ分岐
jsr label	labelから始まる関数の呼出し
mov op1,op2	op1をop2に転送
rts	関数からのリターン
shift op1,op2	op1をop2の値だけ右にビットシフト
spl op1,op2,op3	[第1命令デコーダ103aの場合]: op1のゼロラン長をop2に書き込む 〜 [第8命令デコーダ103hの場合]: アドレスop1から始まるメモリの内容をアドレスop2から始まるメモリにサイズop3分だけブロック転送

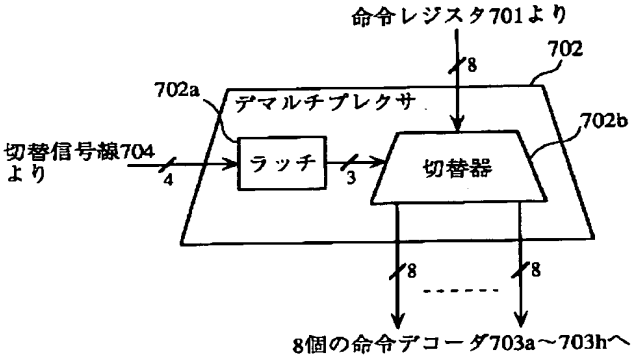
【図６】

オブジェクト	ステップno.
_m	
mov &a, t1	S600
jsr f	S601
mov &a, t1	S602
jsr g	S603
rst	S604
_f	
ex8	S605
spl &b, t1, 10	S606
rst	S607
_g	
exl	S608
mov 0, i	S609
_L1	
spl (&a+i), j	S610
spl (&b+i), k	S611
beq k, j, _L3	S612
jsr _error	S613
_L3	
add 1, i	S614
blt i, 10, _L1	S615
rts	S616

【図７】



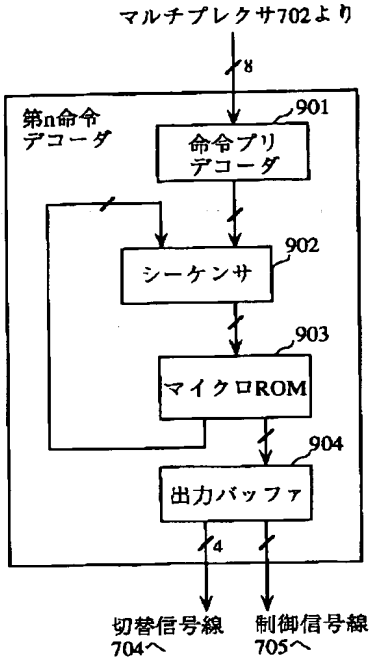
【図８】



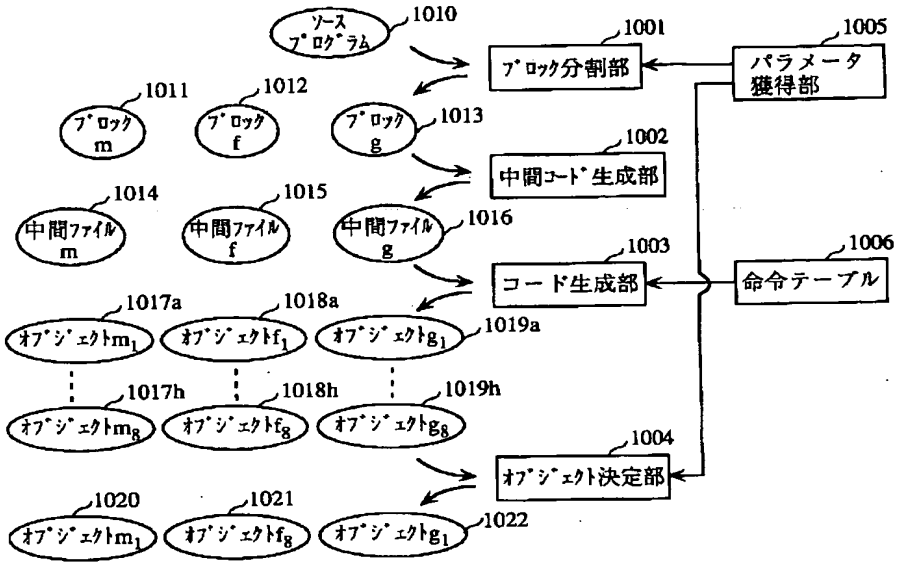
【図１４】

```
_m                  /*関数m ()*/
      mov &a, t1    /配列aのアドレスをt1に転送*/
      jsr f         /*_fを呼び出す*/
      mov &a, t1    /配列aのアドレスをt1に転送*/
      jsr g         /*_gを呼び出す*/
      rts           /*リターン*/
$size = 11 bytes
$cycle = 16
```

【図９】



【図１０】

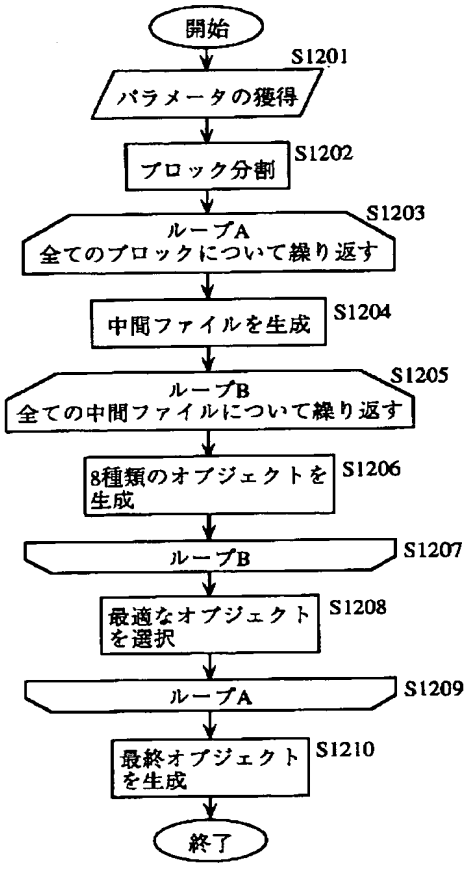


【図１１】

1101 1102

中間コード	命令セット1				命令セット8		
	命令	サイズ	サイクル		命令	サイズ	サイクル
Y=Xadd Y	add x, y	3	5		add x, y	3	5

【図１２】



【図１３】

```
char b[10];    /*グローバル配列bを宣言*/
m()
{
  char a[10];  /*ローカル配列bを宣言*/
  f(&a);       /*配列bを配列aにコピー*/
  g(&b);       /*配列aと配列bの各要素のゼロラン長を比較*/
}
f(char *a)                  /*配列bを配列aにコピー*/
{
  char i;                   /*ローカル変数iを宣言*/
  for(i=0; i<10 ; i++)      /*i=0～9まで繰り返す*/
  {
    a[i] = b[i];            /*b[i]をa[i]に代入*/
  }
}
g(char *a)                  /*配列aと配列bの各要素の最下位ビットから
                              のゼロラン長を比較*/
{
  char i, j, k, t ;         /*ローカル変数i, j, k, tを宣言*/
  for(i=0; i<10 ; i++)      /*i=0～9まで繰り返す*/
  {
    t = a[i];               /*a[i]をtに代入*/
    for(j = 0; 1 & t == 0; t >> 1, j++)  /*tの最下位ビットが1にな
                              るまで、tを右に1ビットシフトし、そのシ
                              フト回数をjとする*/
    t = b[i];               /*b[i]をtに代入*/
    for(k = 0; 1 & t == 0; t >> 1, k++)  /*tの最下位ビットが1にな
                              るまで、tを右に1ビットシフトし、そのシ
                              フト回数をkとする*/
    if(k != j) error();     /*jとkが一致しなければエラー処理*/
  }
}
```

【図１５】

(a)

```
_f                  /*関数f ()*/
      exl           /*第1命令デコーダを指定*/
      mov 0,i       /*ゼロをiに転送*/
_L1
      mov (&b+i),(&a+i)  /*配列データb[i]を配列データa
                           [i]に転送*/
      add 1,i       /*iに1を加算*/
      blt i,10,_L1  /*i<10の場合は_L1に分岐*/
      rts           /*リターン*/
$size = 18 bytes
$cycle = 24
```

(b)

```
_f                  /*関数f ()*/
      ex8           /*第8命令デコーダを指定*/
      spl &b, t1,10 /*アドレスbからサイズ10のメモリ
                      ブロックのデータをアドレスt1から
                      サイズ10のメモリブロックに転送*/
      rts           /*リターン*/
$size = 4 bytes
$cycle = 6
```

【図１６】

(a)

```
_g                      /*関数g ()*/
        exl             /*第1命令デコーダを指定*/
        mov 0,i         /*ゼロをiに転送*/
_L1
        spl (&a+i),j    /*配列データa[i]の最下位ビットから
                          のゼロラン長をjに書き込む*/
        spl (&b+i),k    /*配列データb[i]の最下位ビットから
                          のゼロラン長をkに書き込む*/
        beq k,j,_L2     /*j=kの場合は_L2に分岐*/
        jsr _error      /*エラー処理*/
_L2
        add 1,i         /*iに1を加算*/
        blt i,10,_L1    /*i<10の場合は_L1に分岐*/
        rts             /*リターン*/
$size = 22 bytes
$cycle = 32
```

(b)

```
_g                      /*関数g ()*/
        cx8             /*第8命令デコーダを指定*/
        mov 0,i         /*ゼロをiに転送*/
_L1
        mov (&a+i),t    /*配列データa[i]をtに転送*/
        mov 0,j         /*ゼロをjに転送*/
_L2
        mov t,t2        /*tをt2に転送*/
        and 1,t2        /*t2と1との論理積をt2とする*/
        shift 1,t       /*tを右に1ビットシフト*/
        add 1,j         /*jに1を加算*/
        beq 0,t2,_L2    /*t2=0の場合は_L2に分岐*/
        mov (&b+i),t    /*配列データb[i]をtに転送*/
        mov 0,k         /*ゼロをkに転送*/
_L3
        mov t,t2        /*tをt2に転送*/
        and 1,t2        /*t2と1との論理積をt2とする*/
        shift 1,t       /*tを右に1ビットシフト*/
        add 1,k         /*kに1を加算*/
        beq 0,t2,_L3    /*t2=0の場合は_L3に分岐*/
        beq k,j,_L4     /*j=kの場合は_L4に分岐*/
        jsr _error      /*エラー処理*/
_L4
        add 1,i         /*iに1を加算*/
        blt i,10,_L1    /*i<10の場合は_L1に分岐*/
        rts             /*リターン*/
$size = 46 bytes
$cycle = 62
```